A Practical Multilevel Quasi-Monte Carlo Method for PDEs with Random Coefficients

Pieterjan Robbe
pieterjan.robbe@kuleuven.be

joint work with Dirk Nuyens & Stefan Vandewalle

March 22, 2016

KU Leuven - University of Leuven
Department of Computer Science
Celestijnenlaan 200A
B3001 Leuven, Belgium

Overview

Intro: High-dimensional Problems

Case Study: PDEs with Random Coefficients

Multilevel Monte Carlo

Multilevel Quasi-Monte Carlo

Summary: Key Features of MLMC

High-dimensional problems

Consider the integral

$$I_s(f) \coloneqq \int_{[0,1]^s} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{1}$$

with s the dimension of the problem, typically large

Approximate (1) by a quadrature rule

$$Q_N(f) \coloneqq \sum_{n=0}^{N-1} w_n f(\boldsymbol{x}_n)$$

with weights $w_n \in \mathbb{R}$ and nodes $\boldsymbol{x}_n \in [0,1]^s$

High-dimensional problems

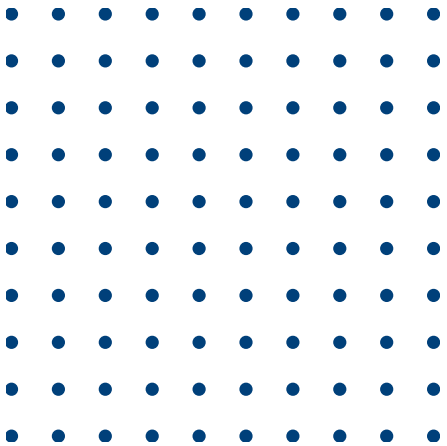
$s = 1$ $\bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet \bullet$ $N = 10$

High-dimensional problems

$s = 1$

$N = 10$

$s = 2$



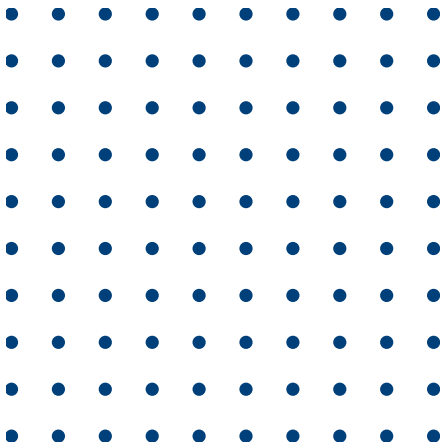
$N = 100$

High-dimensional problems

$s = 1$

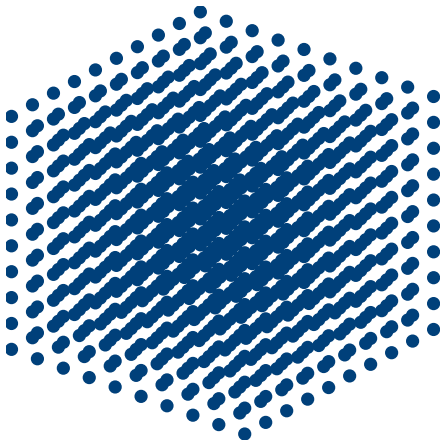
$N = 10$

$s = 2$



$N = 100$

$s = 3$



$N = 1000$

High-dimensional problems

Thus, the amount of work N required to achieve a desired accuracy ϵ grows exponentially with the dimension s,

$$\epsilon(N) = \mathcal{O}(N^{-r/s})$$

where r represents the smoothness of the function f

High-dimensional problems

Thus, the amount of work N required to achieve a desired accuracy ϵ grows exponentially with the dimension s,

$$\epsilon(N) = \mathcal{O}(N^{-r/s})$$

where r represents the smoothness of the function f

Already for moderate dimensions the order of convergence is so slow that the method is of limited practical use. This is

High-dimensional problems

Thus, the amount of work N required to achieve a desired accuracy ϵ grows exponentially with the dimension s,

$$\epsilon(N) = \mathcal{O}(N^{-r/s})$$

where r represents the smoothness of the function f

Already for moderate dimensions the order of convergence is so slow that the method is of limited practical use. This is

On the other hand, randomised algorithms have a work-complexity rate independent of the dimension. For example, the classical Monte Carlo method has

$$\epsilon(N) = \mathcal{O}(N^{-1/2})$$

PDEs with random coefficients

Steady-state flow through porous media can be described by Darcy's law, taking the form of an elliptic PDE with random diffusion coefficient

$$-\nabla \cdot (k(\boldsymbol{x};\omega)\nabla p(\boldsymbol{x};\omega)) = f(\boldsymbol{x})$$

with $\boldsymbol{x} \in [0,1]^3$ and $\omega \in \Omega$

PDEs with random coefficients

Steady-state flow through porous media can be described by Darcy's law, taking the form of an elliptic PDE with random diffusion coefficient

$$-\nabla \cdot (k(\boldsymbol{x};\omega)\nabla p(\boldsymbol{x};\omega)) = f(\boldsymbol{x})$$

with $\boldsymbol{x} \in [0,1]^3$ and $\omega \in \Omega$

The diffusion coefficient is modelled as a lognormal random field with exponential covariance function

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \exp\left(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_p}{\lambda}\right),$$

PDEs with random coefficients

Steady-state flow through porous media can be described by Darcy's law, taking the form of an elliptic PDE with random diffusion coefficient

$$-\nabla \cdot (k(\boldsymbol{x};\omega)\nabla p(\boldsymbol{x};\omega)) = f(\boldsymbol{x})$$

with $\boldsymbol{x} \in [0,1]^3$ and $\omega \in \Omega$

The diffusion coefficient is modelled as a lognormal random field with exponential covariance function

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \exp\left(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_p}{\lambda}\right),$$

or, more generally as a Matérn kernel

$$C(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_p}{\lambda/2\sqrt{\nu}}\right) K_\nu \left(\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_p}{\lambda/2\sqrt{\nu}}\right)$$

PDEs with random coefficients

representation of uncertainty

We use the KL-expansion to take samples from the diffusion coefficient

$$k(\boldsymbol{x};\omega) = \bar{k} + \exp\left(\sum_{n=1}^{\infty} \sqrt{\theta_n} f_n(\boldsymbol{x})\xi(\omega)\right) \tag{2}$$

with θ_n and f_n the eigenvalues and eigenfunctions of the covariance operator $C(\boldsymbol{x}_1, \boldsymbol{x}_2)$

PDEs with random coefficients

representation of uncertainty

We use the KL-expansion to take samples from the diffusion coefficient

$$k(\boldsymbol{x};\omega) = \bar{k} + \exp\left(\sum_{n=1}^{\infty} \sqrt{\theta_n} f_n(\boldsymbol{x})\xi(\omega)\right)$$

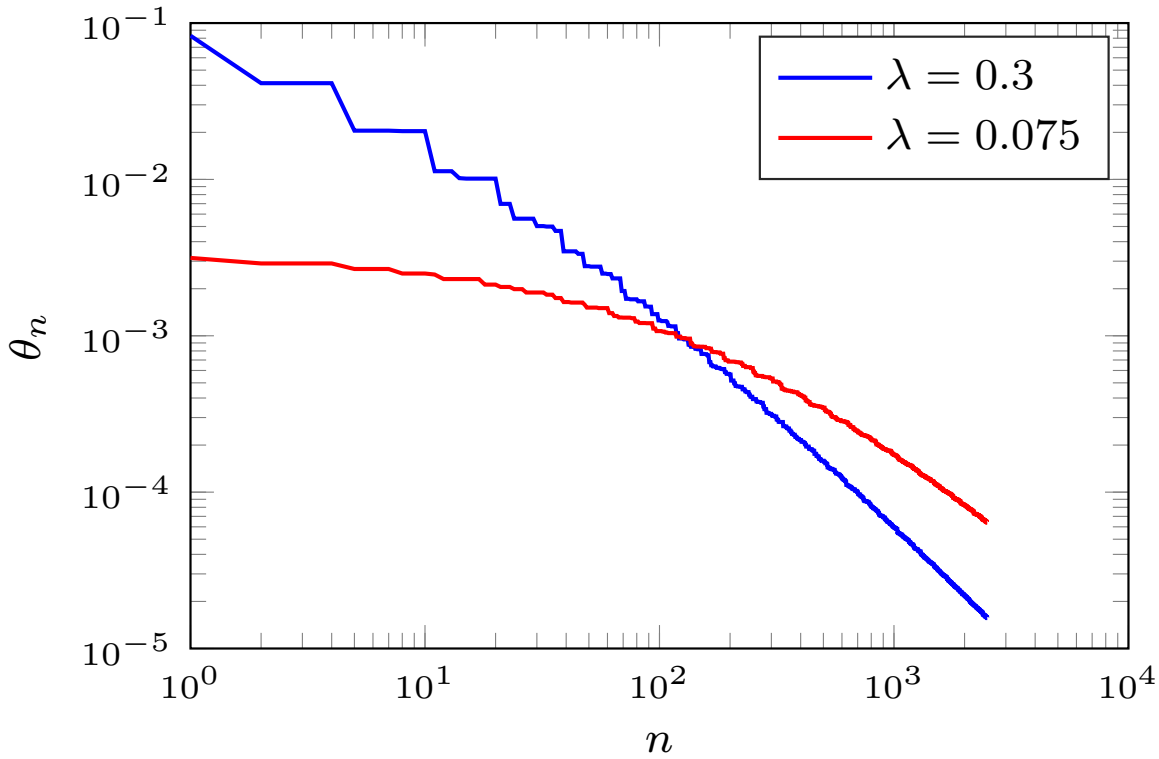
with θ_n and f_n the eigenvalues and eigenfunctions of the covariance operator $C(\boldsymbol{x}_1, \boldsymbol{x}_2)$

- ⋆ We use $p = 1$ (1-norm) in (2) because analytic expressions exist for eigenvalues and eigenfunctions
- ⋆ In practice, the infinite sum must be truncated after s terms

We can also use fast circulant embedding techniques to sample from the random field

PDEs with random coefficients

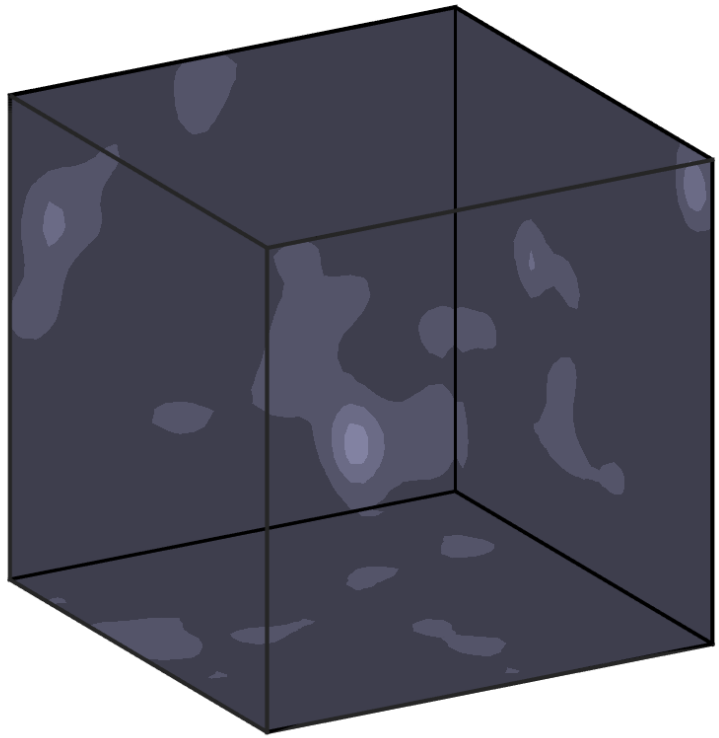
structure of the eigenvalues



Decay of the three-dimensional eigenvalues in the KL-expansion for $\lambda = 0.3$ and $\lambda = 0.075$. In both cases, the variance $\sigma^2 = 1$.

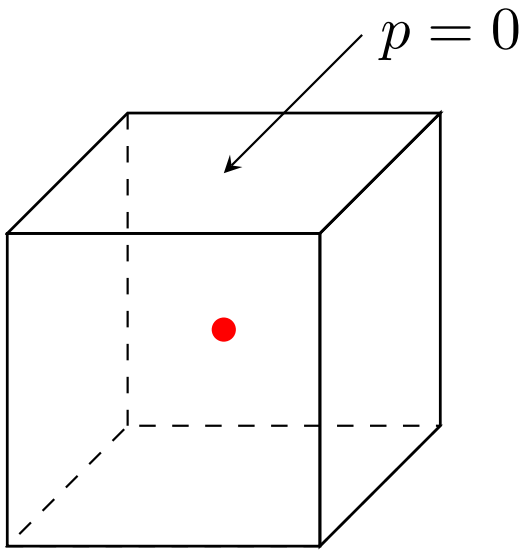
PDEs with random coefficients

a typical sample of the diffusion coefficient

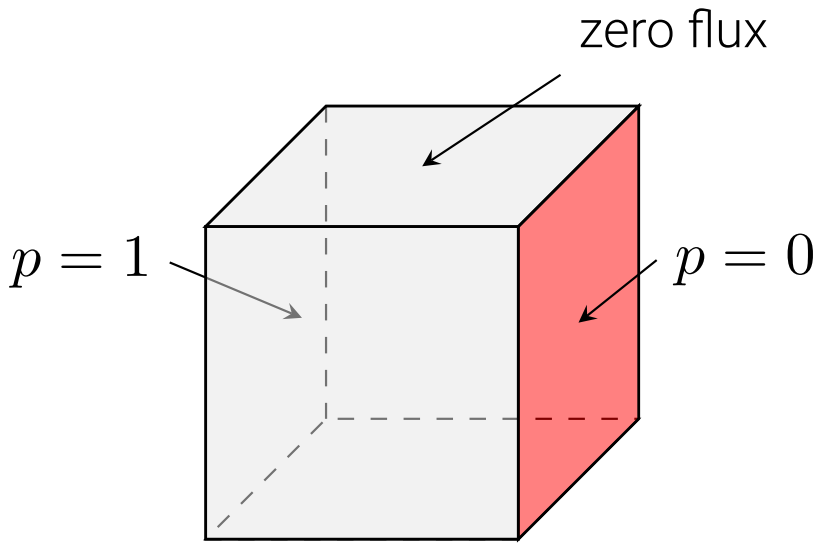


PDEs with random coefficients

quantity of interest



G1

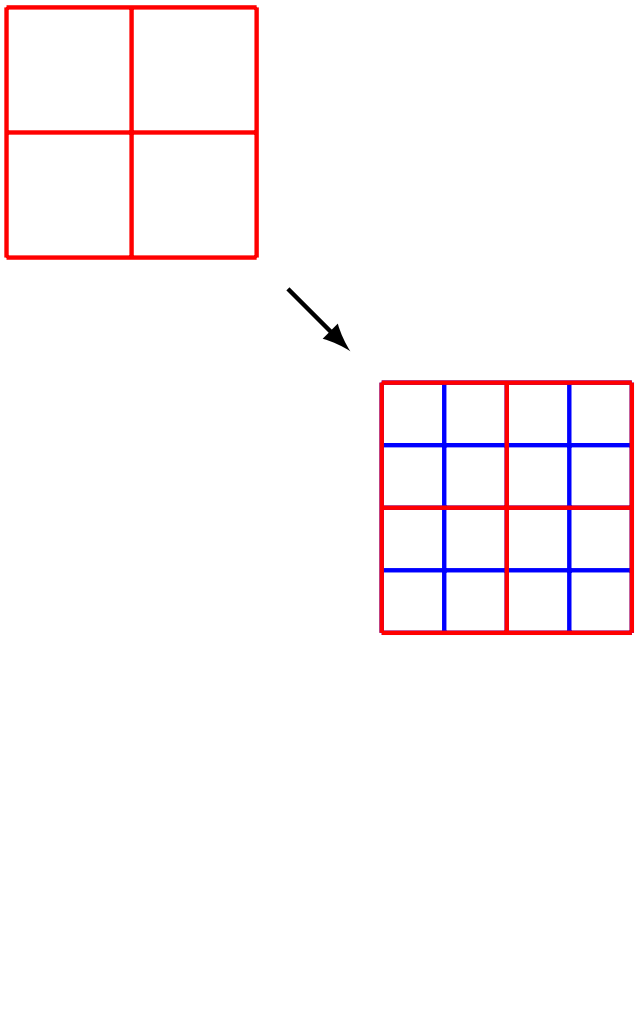


G2

We are interested in the statistics of a quantity of interest $\mathsf{G} := \mathcal{G}(p(\boldsymbol{x}; \omega))$

In particular, what are $\mathbb{E}[\mathsf{G1}]$ and $\mathbb{E}[\mathsf{G2}]$?

Multilevel Monte Carlo



$$\begin{aligned}\mathbb{E}[G_1] &= \mathbb{E}[G_0] + (\mathbb{E}[G_1] - \mathbb{E}[G_0]) \\ &= \mathbb{E}[G_0] + \mathbb{E}[G_1 - G_0] \\ \mathbb{E}[G_2] &= \mathbb{E}[G_1] + \mathbb{E}[G_2 - G_1] \\ &= \mathbb{E}[G_0] + \mathbb{E}[G_1 - G_0] + \mathbb{E}[G_2 - G_1]\end{aligned}$$

$$\mathbb{E}[G_L] = \sum_{\ell=0}^{L} \mathbb{E}[\Delta G_\ell]$$

Multilevel Monte Carlo

Let $\Delta\mathcal{Q}_\ell$ be an unbiased estimator for $\mathbb{E}[\Delta G_\ell]$, then we define the multilevel estimator as

$$\mathcal{M} := \sum_{\ell=0}^{L} \Delta\mathcal{Q}_\ell \approx \mathbb{E}[G]$$

Multilevel Monte Carlo

Let $\Delta\mathcal{Q}_\ell$ be an unbiased estimator for $\mathbb{E}[\Delta G_\ell]$, then we define the multilevel estimator as

$$\mathcal{M} := \sum_{\ell=0}^{L} \Delta\mathcal{Q}_\ell \approx \mathbb{E}[G]$$

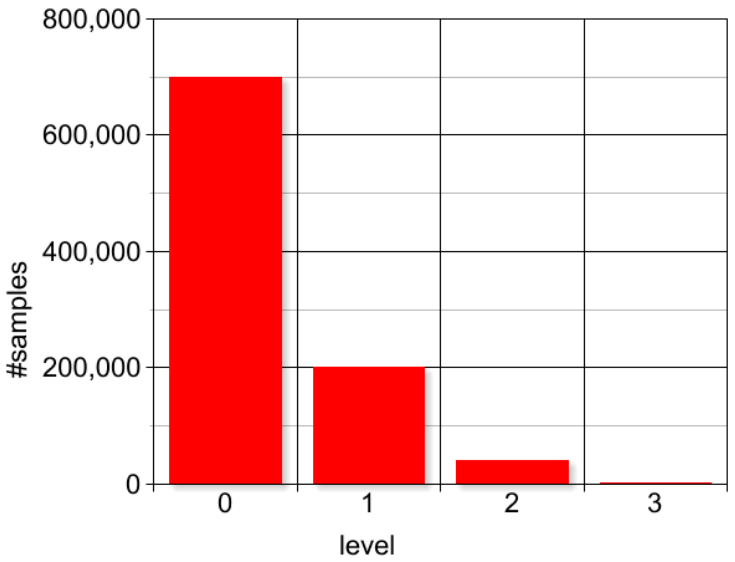
When using a Monte Carlo estimator for each of the contributions $\Delta\mathcal{Q}_\ell$, we obtain a multilevel Monte Carlo (MLMC) estimator for $\mathbb{E}[G]$:

$$\boxed{\mathcal{M}_L = \sum_{\ell=0}^{L} \frac{1}{N_\ell} \sum_{n=0}^{N_\ell - 1} \Delta G_\ell(\xi_n)}$$

Multilevel Monte Carlo

By performing an error analysis of the multilevel estimator it is possible to obtain an expression for the optimal number of samples at each level[1]

$$N_\ell = \mathrm{TOL}^{-2} \sqrt{\frac{V_\ell}{W_\ell}} \sum_{m=0}^{L} \sqrt{V_m W_m} \quad (2)$$

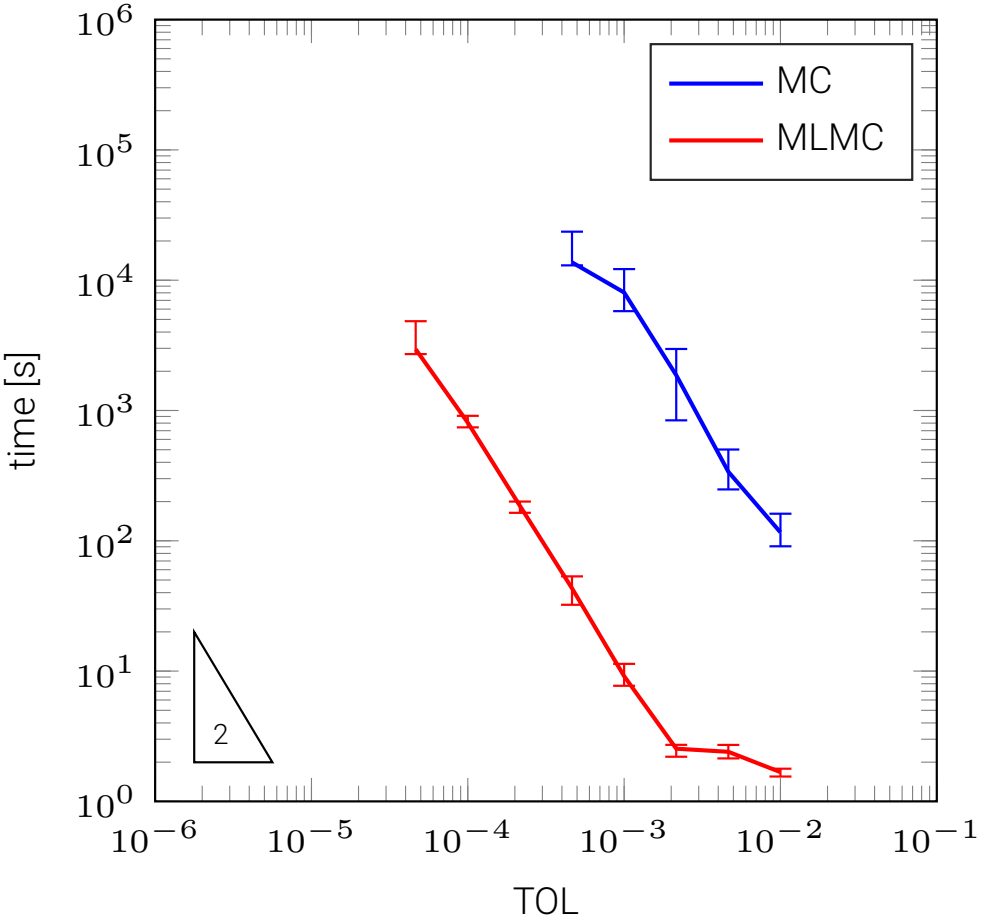


This expression can be used in a level-adaptive algorithm: starting from a coarsest mesh, estimate the bias and add a finer level when needed. On each level, take the optimal amount of N_ℓ samples

[1] Giles, Michael B. "Multilevel Monte Carlo path simulation." *Operations Research*, 56.3 (2008): 607-617.

Performance of MLMC

results for G1



Simulation details:

- ⋆ correlation length $\lambda = 0.3$, variance $\sigma^2 = 1$
- ⋆ $s = 250$ uncertainties
- ⋆ finest grid has $4 \cdot 2^5$ grid points in each dimension
- ⋆ the PDE is solved using a finite volume method
- ⋆ the sparse solver is CG with multigrid preconditioning as implemented in `hsl_mi20`

Multilevel Quasi-Monte Carlo

The Quasi-Monte Carlo (QMC) method is an equal-weight cubature rule:

$$I_s(f) = \int_{[0,1]^s} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \approx \mathcal{Q}_{s,N}(f) := \frac{1}{N} \sum_{n=0}^{N-1} f(\boldsymbol{t}_n)$$

Multilevel Quasi-Monte Carlo

The Quasi-Monte Carlo (QMC) method is an equal-weight cubature rule:

$$I_s(f) = \int_{[0,1]^s} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \approx Q_{s,N}(f) := \frac{1}{N} \sum_{n=0}^{N-1} f(\boldsymbol{t}_n)$$

A popular choice for $\boldsymbol{t}_n$ are so called rank-1 lattice rules

$$\boldsymbol{t}_n = \left\{ \frac{n\boldsymbol{z}}{N} \right\}, \quad n = 0, \ldots, N-1,$$

where $\boldsymbol{z} \in \mathbb{Z}^s$ is a generating vector, and $\{ \cdot \}$ denotes the fractional part

Multilevel Quasi-Monte Carlo

The Quasi-Monte Carlo (QMC) method is an equal-weight cubature rule:

$$I_s(f) = \int_{[0,1]^s} f(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \approx Q_{s,N}(f) := \frac{1}{N} \sum_{n=0}^{N-1} f(\boldsymbol{t}_n)$$

A popular choice for $\boldsymbol{t}_n$ are so called rank-1 lattice rules

$$\boldsymbol{t}_n = \left\{ \frac{n\boldsymbol{z}}{N} \right\}, \quad n = 0, \ldots, N-1,$$

where $\boldsymbol{z} \in \mathbb{Z}^s$ is a generating vector, and $\{ \cdot \}$ denotes the fractional part

For integrands with sufficient smoothness and progressively less important dimensions, there exists lattice rules for which

$$\epsilon(N) = \mathcal{O}(N^{-1}(\log N)^s)$$

Multilevel Quasi-Monte Carlo

the problem of random shifting

The classical MC method comes with a probabilistic error bound of the form $\sigma(f)/\sqrt{N}$, where $\sigma^2 := I_s(f^2) - (I_s(f))^2$ is the variance of f

Unfortunately, QMC methods do not immediately provide such an error bound, since the points are chosen deterministically

Multilevel Quasi-Monte Carlo

the problem of random shifting

The classical MC method comes with a probabilistic error bound of the form $\sigma(f)/\sqrt{N}$, where $\sigma^2 := I_s(f^2) - (I_s(f))^2$ is the variance of f

Unfortunately, QMC methods do not immediately provide such an error bound, since the points are chosen deterministically

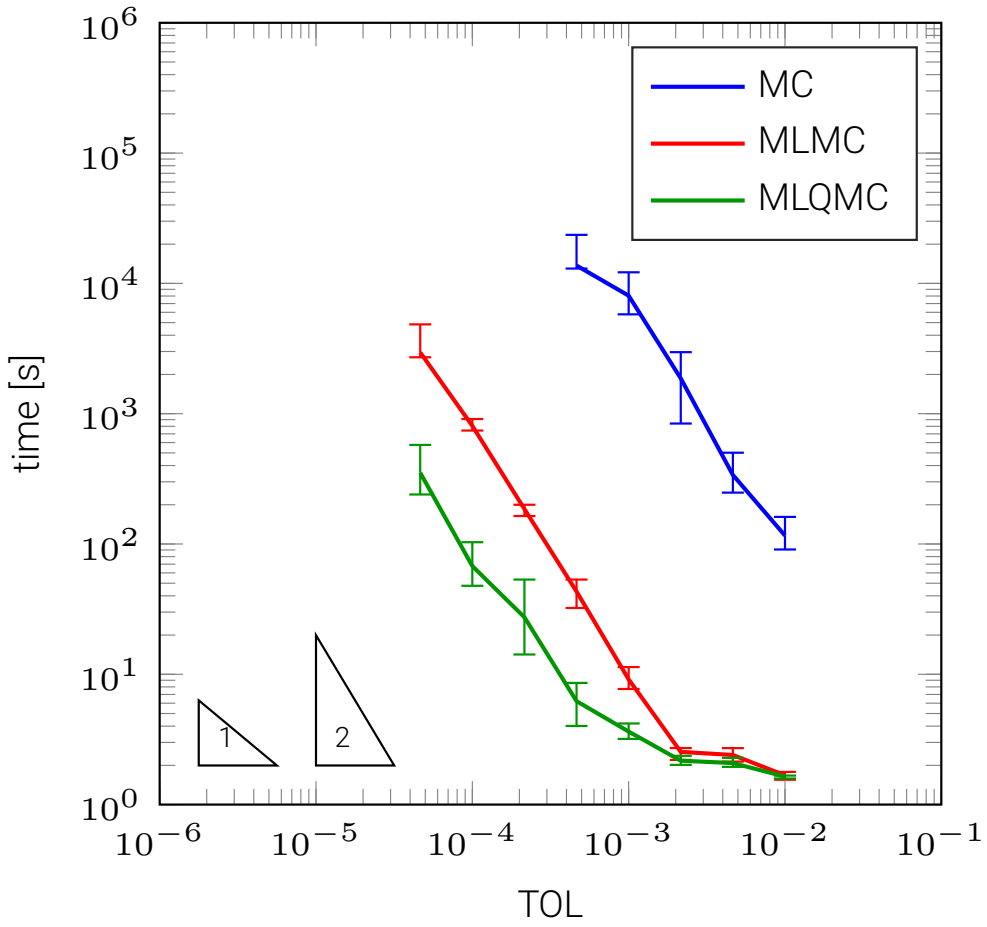
The feature can be recovered by using $\mathcal{U}(0,1)$-distributed random shifts:

$$\boldsymbol{t}_n = \left\{ \frac{n\boldsymbol{z}}{N} + \boldsymbol{\Delta} \right\}, \quad n = 0 \dots N-1$$

One takes K different random shifts, and the variance $\mathbb{V}_{\boldsymbol{\Delta}}$ over these K estimators can be used to construct a confidence interval

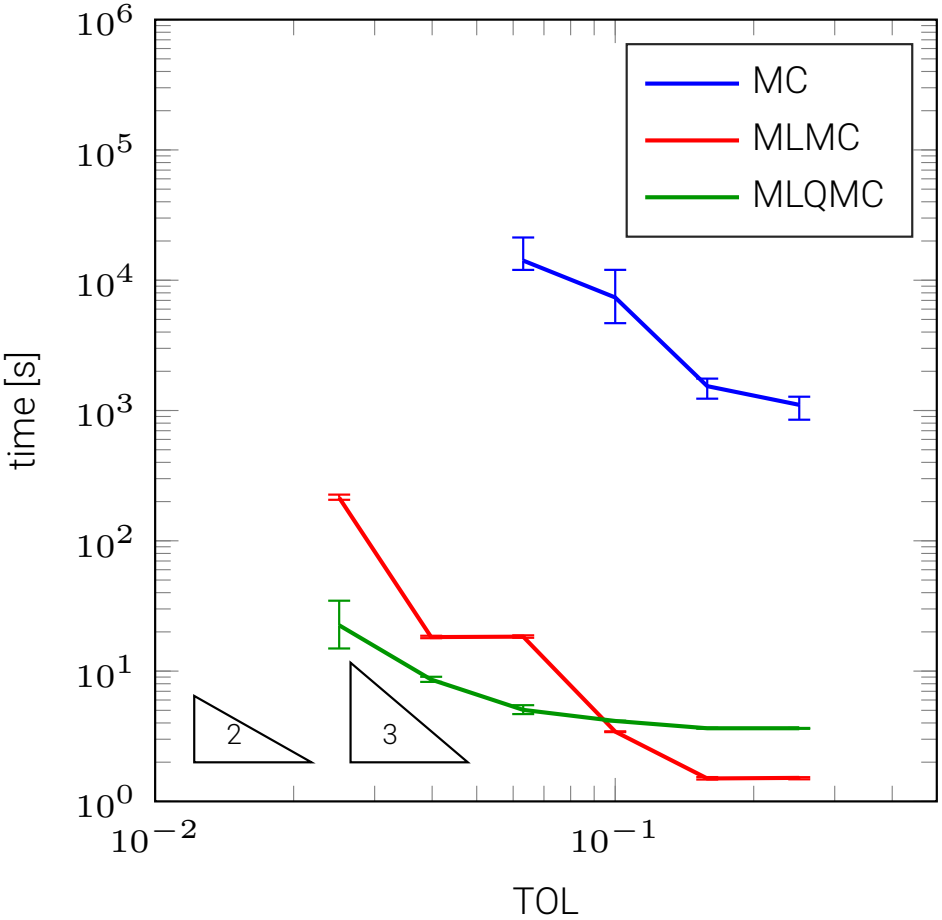
Performance of MLQMC

results for G1



Performance of MLMC

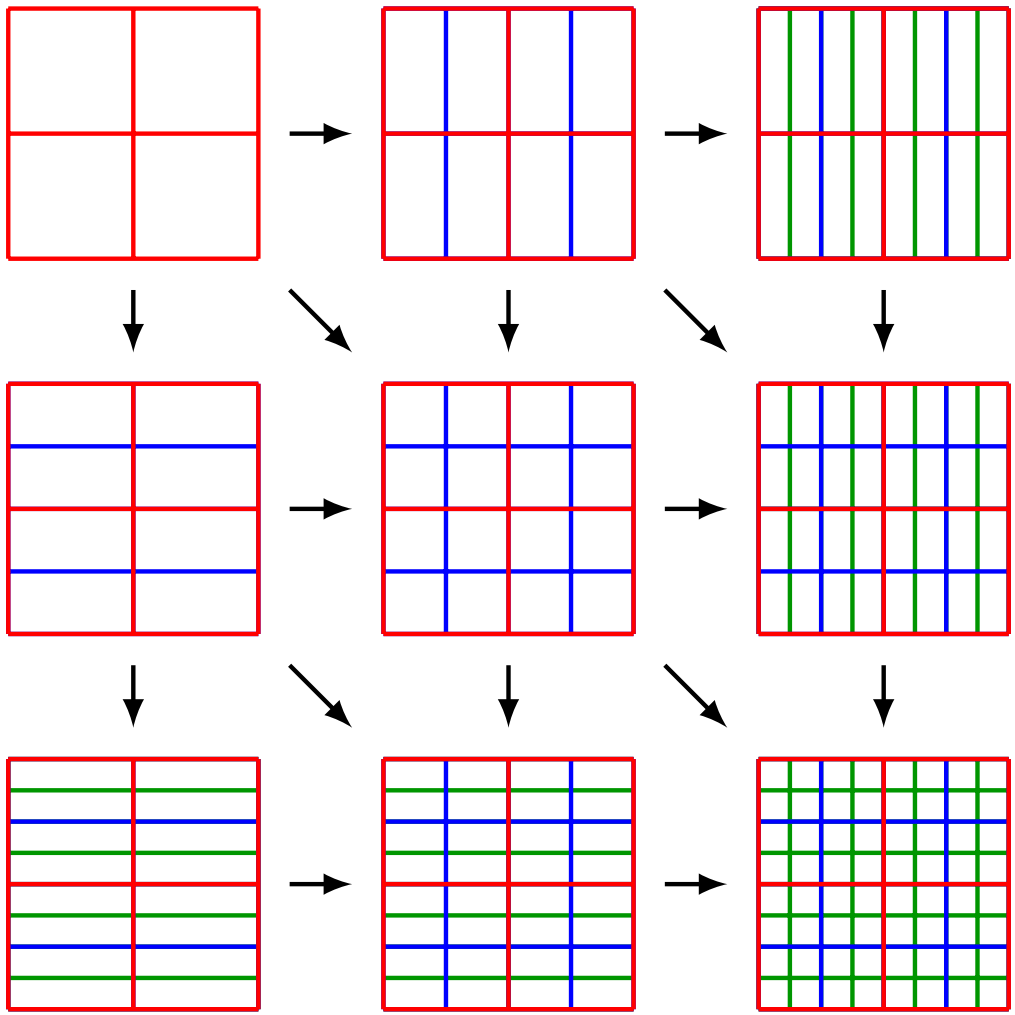
results for G2



Simulation details:

- ⋆ correlation length $\lambda = 0.075$, variance $\sigma^2 = 1$
- ⋆ $s = 2500$ uncertainties
- ⋆ finest grid has $4 \cdot 2^5$ grid points in each dimension
- ⋆ the PDE is solved using a finite volume method
- ⋆ the sparse solver is CG with multigrid preconditioning as implemented in `hsl_mi20`

Multi-Index Monte Carlo



Multi-Index Monte Carlo

Define the difference operator along a single direction i, denoted Δ_i, by

$$\Delta_i = \begin{cases} G_{\boldsymbol{\ell}} - G_{\boldsymbol{\ell}-\boldsymbol{e}_i} & \text{if } \boldsymbol{e}_i \cdot \boldsymbol{\ell} > 0 \\ G_{\boldsymbol{\ell}} & \text{if } \boldsymbol{e}_i \cdot \boldsymbol{\ell} = 0 \end{cases},$$

where $\boldsymbol{e}_i$ is the unit vector in direction i

Next, we define the difference operator $\Delta = \prod_{i=1}^{d} \Delta_i$

Again, the expected value can be expressed as the telescoping sum

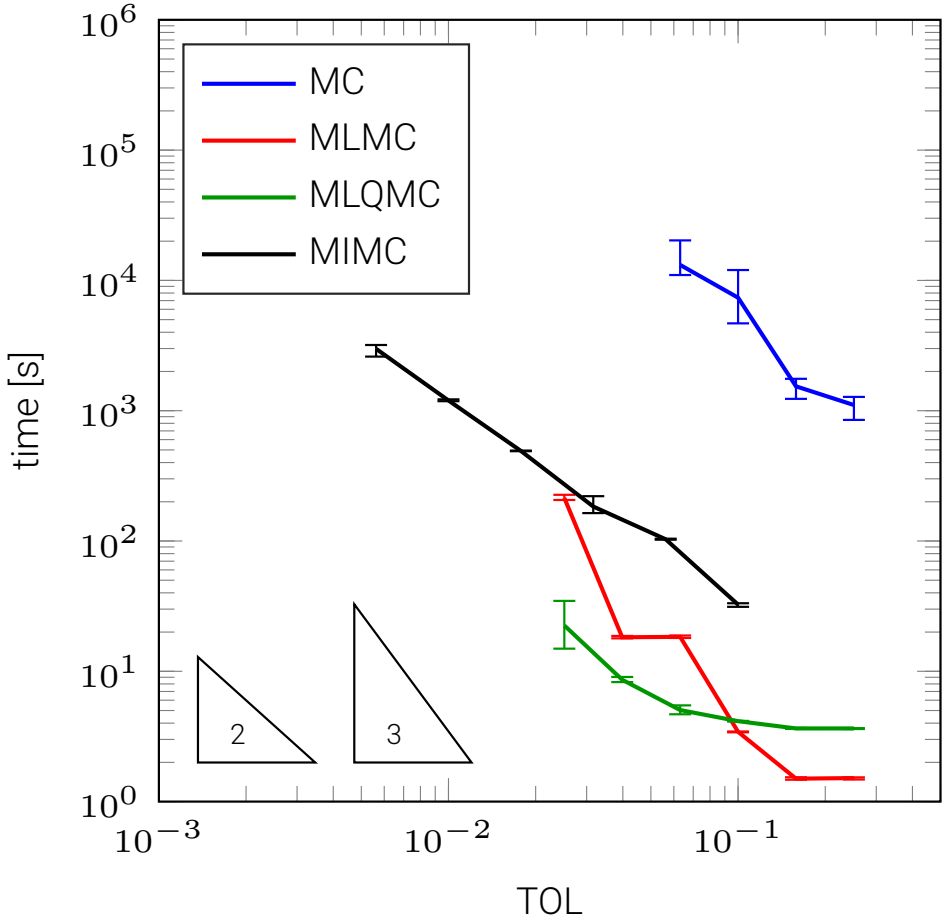
$$\mathbb{E}[G] = \sum_{\boldsymbol{\ell} \geq 0} \mathbb{E}[\Delta G_{\boldsymbol{\ell}}]$$

By choosing a suitable subset[2] of all $\boldsymbol{\ell}$ we can reduce the bias of the estimator and avoid to take samples at $\boldsymbol{\ell} = (L, L, L)$

[2] Haji-Ali, Abdul-Lateef, Fabio Nobile, and Raúl Tempone. "Multi-index Monte Carlo: when sparsity meets sampling." *Numerische Mathematik* (2015): 1-40.

Performance of MIMC

results for G2



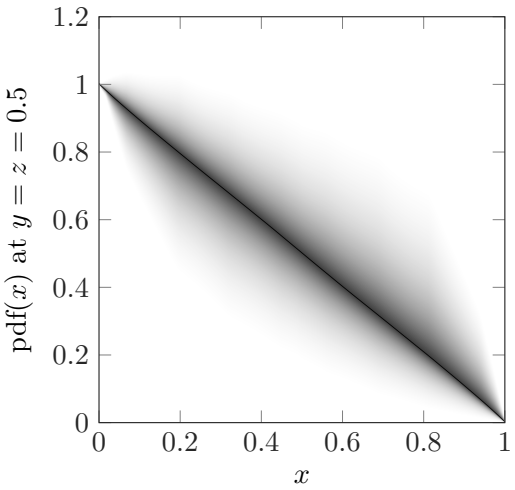
Summary: Key Features of ML(Q)MC/MIMC

(i) embarrassingly parallel problem

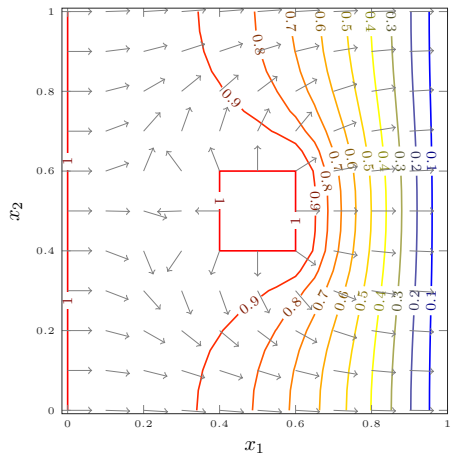
(ii) obtain probability density function of quantity of interest

(iii) multiple quantities of interest in single simulation

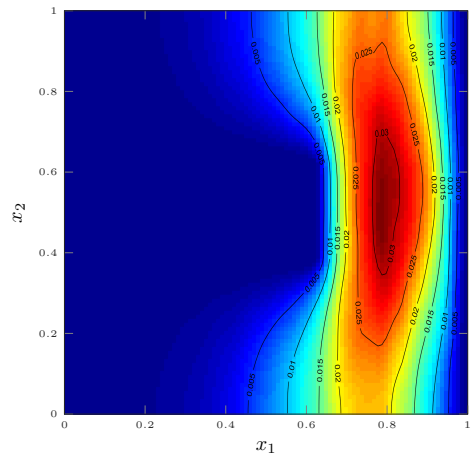
(iv) not limited to unit square domain: complex geometries



feature (ii) & (iii)



features (iii) & (iv)

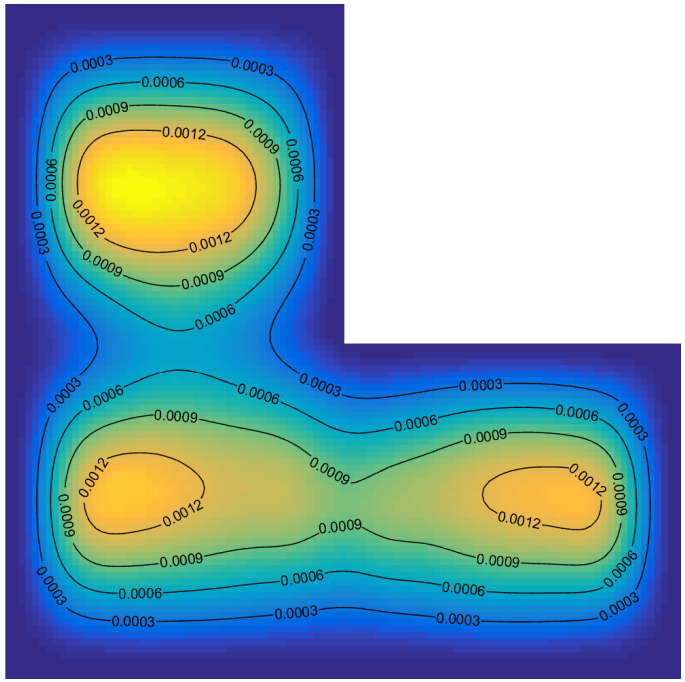


features (iii) & (iv)

Questions?

Some suggestions. . .

- ⋆ How does the MLMC method compare to other methods for high-dimensional integration, such as sparse grids?
- ⋆ Wait - can you tell me more about quasi-Monte Carlo (QMC)?
- ⋆ How do you arrive at this formula for the optimal number of samples?
- ⋆ But how does the MLMC algorithm actually work?



Comparison with sparse grids

Classical sparse grids have

$$\epsilon(N) = \mathcal{O}(N^{-r}(\log N)^{(s-1)(r+1)})$$

for $r > 1$, this is better than QMC(*)

Hence, performance depends on regularity and number of dimensions

In general, sparse grids perform better for smooth problems with low number of dimensions

back

Optimal number of samples

The mean-square error (MSE) of $\mathcal{M}$ can be expanded as

$$\begin{aligned}\mathrm{MSE}(\mathcal{M}) &= \mathbb{E}[(\mathcal{M} - \mathbb{E}[\mathcal{M}])^2] + (\mathbb{E}[\mathcal{M}] - G)^2 \\ &= \mathbb{V}[\mathcal{M}] + \mathrm{Bias}(\mathcal{M}, G)^2\end{aligned}$$

We bound both terms as

$$\mathrm{Bias}(\mathcal{M}, G) \leq (1-\theta)\epsilon, \text{ and} \qquad \text{(bias constraint)}$$

$$\mathrm{prob}[|\mathcal{M} - \mathbb{E}[\mathcal{M}]| \leq \theta\epsilon] \geq 1 - \nu, \qquad \text{(statistical constraint)}$$

where ϵ is a tolerance, θ the error splitting and ν a failure probability

Note that, by normality of the estimator, we can rewrite the statistical constraint as

$$\mathbb{V}[\mathcal{M}] \leq (\theta \mathrm{TOL})^2 \quad \text{with} \quad \mathrm{TOL} := \frac{\epsilon}{\Phi^{-1}(1-\nu/2)}$$

Optimal number of samples

The optimal number of samples N_ℓ can be obtained from

$$\min_{N_\ell} \text{ Total Work} = \sum_{\ell=0}^{L} N_\ell W_\ell$$
$$\text{s.t.} \quad \mathbb{V}[\mathcal{M}_L] \leq (\theta \text{TOL})^2$$

with W_ℓ the amount of work to compute a single realisation of ΔG_ℓ

For MLMC, we have that $\mathbb{V}[\mathcal{M}_L] = \sum_{\ell=0}^{L} \frac{V_\ell}{N_\ell}$

For MLQMC, we find $\mathbb{V}[\mathcal{M}_L] \lesssim \sum_{\ell=0}^{L} \frac{V_\ell}{K N_\ell^{-2}}$

back

MLMC algorithm

begin

  $L := -1;\ \theta := 0.5;$ converged := **false**;

  repeat

    $L \leftarrow L + 1;$

    take N^* samples at level L and compute sample variance and bias;

    if *bias* $< \epsilon/2$ **then**

      $\theta \leftarrow 1 - \text{bias}/\epsilon;$

    end

    compute optimal number of samples at each $\ell < L$ with (2);

    update samples at each $\ell < L$;

    if $L > 2$ **then**

      recompute bias and check for convergence;

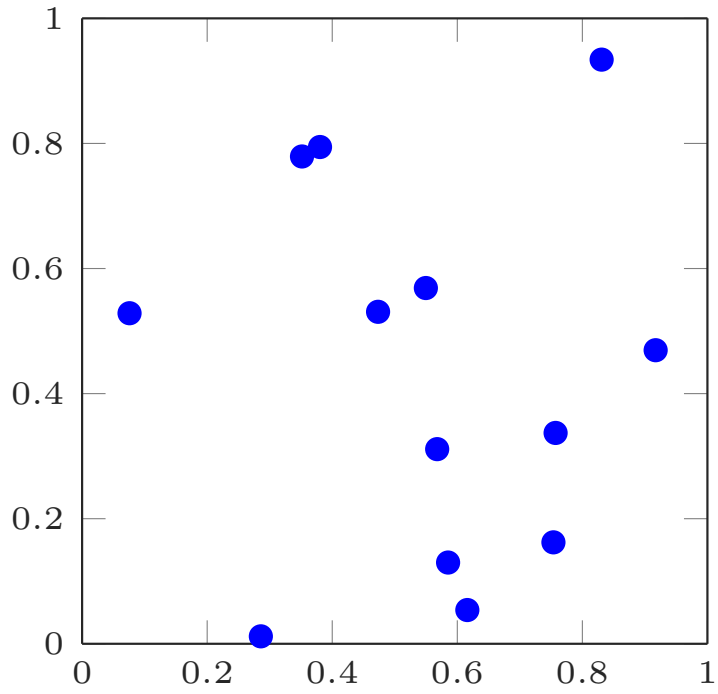
    end

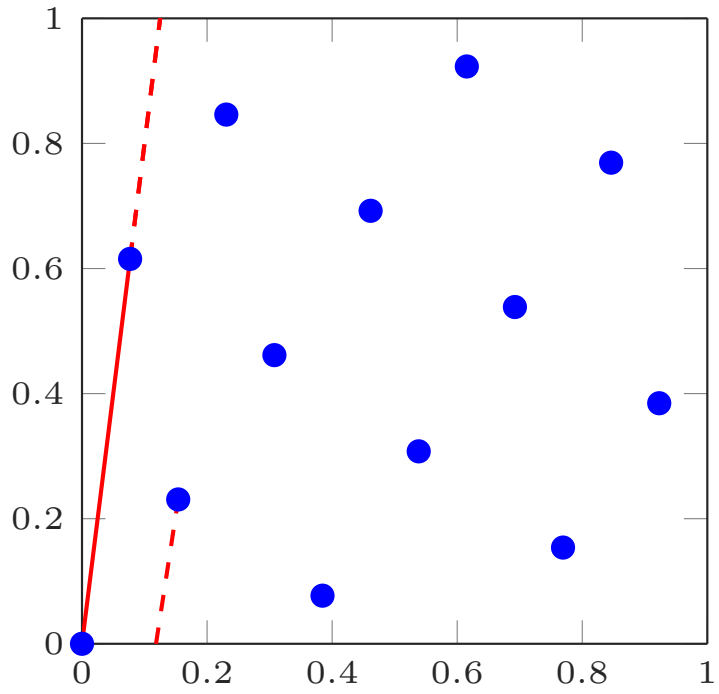
  until *converged* = **true**;

end

back

Quasi-Monte Carlo (QMC)





back